US-PAT-NO: 6311175

DOCUMENT-IDENTIFIER: US 6311175 B1

TITLE: System and method for generating performance models of complex information technology systems

---------- KWIC ---------

Brief Summary Text - BSTX (6):

Although interconnected systems, such as the one shown in FIG. 1, offer many advantages to the users, e.g., **resource** sharing, as such systems grow and the number of component interlinkages increase, the behavior of these complex systems becomes more difficult to predict. Further, system performance begins to lag or becomes inconsistent, even becoming chaotic in nature. The addition or removal of one component, even seemingly minor, could have dramatic consequences on the performance of the whole system. Even an upgrade on one component could adversely affect a distant, seemingly unrelated component. The system and method of the present invention is directed to techniques to better predict the behavior of complex IT systems, offering system administrators the opportunity to identify problem areas such as performance bottlenecks and to correct them prior to a system or component failure.

Brief Summary Text - BSTX (13):

As discussed, multiple applications will be operated within a common IT infrastructure, such as the one shown in FIG. 1. Often, these applications will utilize some of the same **resources**. It is obvious that the sharing of IT infrastructure **resources** among different applications may cause unexpected interactions on system behavior, and that often such unexpected interactions, being non-synergistic, are undesirable. An example would be multiple business applications sharing a router within an IT system. As illustrated, a particular application, e.g., an E-mail service, burdens a router in such a way that other applications do not function well. In this example, it is reasonable to expect numerous applications to, at times, share usage of the router. Traditional systems management techniques may prove difficult in determining which specific application is causing loss of system performance. This example further explains why there is a need to find hidden relationships among IT system components and applications running in such environments. By way of solving the problem in this example, it may be necessary to reroute E-mail traffic through another router to obtain adequate performance for the other applications.

Brief Summary Text - BSTX (15):

When system monitoring is included in the aforementioned traditional management system, this monitoring is usually limited to real time data, such as the current system load and the like. An administrator may observe such **reporting** of real time data, and if system loads or events being monitored are

found to be consistent with loads that the administrator recognizes to be associated with impending system malfunction or loss of performance, that administrator may redirect part of the load through alternative subsystems of the IT infrastructure to avert problems.

Brief Summary Text - BSTX (16):

Often, such real time data **reporting** may be used in coordination with a system model of the IT system, of which data is being collected and reported. The model usually includes a computer algorithm that utilizes code governing the relations among various system devices. A problem with such models, however, is that the relations used in modeling the system account only for expected interactions among components and subsystems. The model is, therefore, merely an idealized model of the actual system. Hidden or unexpected relations that exist between components would not be accounted for. Furthermore, as the infrastructure 220 is modified, the model must be manually altered to include new relations in the model algorithm to account for the changes made.

Brief Summary Text - BSTX (18):

Typically, expert systems are developed so that knowledge may be accumulated from a person or persons skilled in a specific area of technology and stored in an easily retrievable media. This way, persons less skilled than the experts, whose knowledge was accumulated within the expert system, have access to such expert information. In this manner, a company may save human and financial **resources** by having less skilled personnel access such expert systems instead of requiring the expert to handle all of such situations requiring a certain level of knowledge.

Brief Summary Text - BSTX (27):

The present invention is directed to a system and method for automatically creating performance models of an information technology (IT) system by use of a continuity analysis, preferably in conjunction with data mining techniques. Adaptive system management is defined as the realization of proactive system management with adaptive techniques that automatically create models of the system and that can learn to plan and predict the effects of management actions in order to meet the various user requirements. IT **Service Level Agreements (SLAs)**, or performance requirements, are predefined constraints or thresholds placed on the system. Performance monitoring of the system is then implemented, from which databases of system state information are determined and stored.

Brief Summary Text - BSTX (28):

A continuity analysis is then performed on the IT system or subsystem thereof by synchronizing **SLA** performance simulations with system monitoring activity, and accumulating both in a historical database. A model of the system environment is then used as input for the continuity analysis. The environment may be defined with any level of detail and is not necessarily a complete or consistent model of the actual system. The system and method of the present invention is preferably implemented with a collection of data

monitors placed throughout the system. These monitors periodically check the state of various elements of the system, storing the monitored data in a database.

Brief Summary Text - BSTX (29):

A test program is then executed, with execution being synchronized with relative monitoring activity, to simulate specific IT system actions related to a specific predefined **SLA**. Execution of the test programs, and the monitoring activities, are preferably performed automatically and at fixed intervals of time. Results of the test program are time measurements of the **SLA**-related actions, which are preferably expressed as real numbers, and which are stored in a database with a time stamp and corresponding monitored system data or equivalently, in an array type data storage scheme. Additional input includes the **SLAs** themselves. These thresholds are used to convert the real numbers from the test program into Boolean values, these Boolean values indicating whether or not the predefined threshold was exceeded or not. This Boolean information is then output to characterize the influence of the various **monitor** values on the targeted performance variable, or the **SLA**. This information may then be used in a number of ways, including trend analysis, performance optimization, and **monitor** optimization.

Detailed Description Text - DETX (3):

FIG. 3 shows a model of an adaptive system management scenario 300 in accordance with the system and method of the present invention. The application of data mining on an information technology (IT) system, such as the one shown in FIG. 1 and generally designated by the reference numeral 305, is illustrated in FIG. 3, in which the IT system 100/305 is connected to at least one **monitor** 310 which monitors the performance of the IT system 305. The **monitor** 310 is connected to a historical database 315, which is used to store various performance measurements on the IT system 100. The historical database 315, in turn, is connected to a number of learning algorithms 320. Elements or events relating to the IT system or infrastructure 305 are monitored throughout the system by appropriate monitoring schemes housed within the monitors 310.

Detailed Description Text - DETX (6):

In devising such a dynamic learning model as disclosed in the present invention, it is first necessary to define thresholds for various system performances. These thresholds are hereinafter referred to as **service level agreements or SLAs,** which in the present invention are simply a numerical threshold used to evaluate a particular performance level of any number of system components or elements. The **SLAs** serve to convert numerical formatted data that is monitored into Boolean values indicating whether the **SLA** threshold was met or not.

Detailed Description Text - DETX (7):

As an example of such an **SLA,** reference is now made to a database 105 in FIG. 1 which is resident on a system server 110 such that numerous and diverse users may query the database 105. In querying database 105, it is reasonable that a login must be first performed through the server 110. This login,

however, may conventionally be performed through another server and database, which is shown in FIG. 1 as server 120 and database 115, respectively. Therefore, for a system user to remotely query database 105, login is first executed through database 115, which upon a successful login grants the user rights to query database 105. For this entire operation, a performance threshold may be established by knowledgeable management personnel, designated in FIG. 2 by the reference numeral 230. Typically, such a threshold would be formed with knowledge of the server 110 and 120 performances on which databases 105 and 115, respectively, reside and a general knowledge of data traffic through these servers.

Detailed Description Text - DETX (8):

For this example, assume the startup time of database 105 is a reasonable **measure** of the performance of database 105. Therefore, the targeted performance level of database 105, i.e., its **SLA**, could be constructed from the access times of both databases 105 and 115. Here, the **SLA** may be delineated as SLA.sub.A where A represents database 105. Since access time has been assumed to be a good **measure** of performance of such an application, total access time for database 105 includes the access time of database 115 since effective execution of database 105 is prolonged by the execution of database 115 which is also referred to herein by the reference indicator B. For this case, the total access time, AT.sub.AB, for the startup of database 105 may be found from the sum of the startup times of the individual databases, AT.sub.A and AT.sub.B, in other words,

Detailed Description Text - DETX (9):

Assume that the study of the individual applications and hardware from which execution of these applications are executed indicates that it is reasonable for the execution of database 115 to take place in no more than 1 second and subsequent execution of database 105 in no more than 2 seconds. From this information, the target for total startup time of database 105, AT.sub.AB, would be for the execution of database 105 in no longer than 3 seconds. This threshold for execution of database 105, including the required access time of database 115, could then be defined for the **SLA of database** 105, hereinafter designated as SLA.sub.AB. This **SLA** would appropriately be recorded as:

Detailed Description Text - DETX (10):

This **SLA** would indicate, in a Boolean format, that execution of database 105 in a time of less than or equal to 3 seconds is satisfactory, e.g., a logical one, and an execution time exceeding 3 seconds is unsatisfactory, e.g., a logical zero. Alternatively, individual thresholds may be defined for databases 105 and 115 and a threshold for overall performance of database 105 obtained by simply summing the individual thresholds, as follows:

Detailed Description Text - DETX (11):

In defining such thresholds, it should be apparent that the greater the number of **SLAs and monitors** 310, shown in FIG. 3, monitoring the IT system 100, shown in FIG. 1, the better the system may be evaluated. Ideally, the majority of IT system 100 components would have **SLAs** associated with them.

Realistically, however, extensive system monitoring presents logistical problems, generally resulting in simpler rather than more complicated models. Nonetheless, as is apparent to those skilled in the art, the greater the number of **SLAs** that may be defined and implemented within the IT system 100, the greater the accuracy of the system model and technique of the present invention in monitoring system performance.

Detailed Description Text - DETX (12):

In order to apply the aforementioned data mining techniques and learning algorithms to historical data on the IT system 100, it is first necessary to build the aforementioned historical database 315, as shown in FIG. 3. It has been determined that the most advantageous method of storing such data is in a conventional relational database format. Typically, all monitored data from the monitors 310 are directed to one central storage location, i.e., the historical database 315. It should be understood, however, that each **monitor** 310 may have its own local memory 330 for storing the monitoring data temporarily, e.g., over a minute, hour, etc., and then later sent to the central historical database 315 where the aforementioned data mining applications may be used to analyze the data.

Detailed Description Text - DETX (13):

It should be understood that the data monitors 310 may be placed throughout the IT infrastructure 100/305 at various components within the system. Monitoring activity may be directed to any number of components, applications or other **resources** with, in general, the overall effectiveness of the present invention enhanced with a corresponding increase in the number of monitors 310 being utilized. These monitors 310 preferably perform their specific monitoring activity automatically and at specific time intervals, collecting data periodically, e.g., once every minute, ten minutes, hour, etc. The type of data being monitored and stored in the historical database 315 may be generally described as state or usage information on a component level, e.g., a harddisk, database, server or other network segment such as the components shown in FIG. 1. For instance, a **monitor** 310 used to **monitor** and record historical data on a particular harddisk may record the free capacity of the disk and whether the disk is being accessed or not. Similar data collected from monitoring a database may include the number of users accessing the database, query volume, and access time.

Detailed Description Text - DETX (14):

In order to perform the continuity analysis on the system 100, it is necessary to evaluate specific system functions over set and defined intervals. For this reason, test programs are utilized to evaluate whether the system 100 is performing within one or more of the aforementioned **SLAs**. For example, it would not be effective to **measure** and evaluate a specific action against its **SLA** only when that action is taken by a person on the network. Such actions would most likely occur pseudo-randomly and would, therefore, not give good indications of the overall performance of the system 100 with respect to time.

Detailed Description Text - DETX (15):

To evaluate the system more effectively, test programs are used to simulate those functions that have associated **SLAs**. In utilizing test programs at defined moments in time, continuity analyses may be performed on the test and monitored data as functions of time. For example, in the case of the **SLA** used for the startup times of databases 105 and 115, a test program would be set up on the server side of the network 100 to simulate a query to these databases. This test program would preferably be executed automatically and at fixed intervals of time. Furthermore, this test program would be substantially synchronized with monitoring events related to the evaluation of the corresponding **SLAs**.

Detailed Description Text - DETX (16):

For the example of SLA.sub.AB as previously defined, a test program to simulate the startup of database 105, with the inclusive startup of database 115, is required. It should be understood that the test program may be executed on the server or client side, the preference being to have the test program executed on both sides. Executing the test program on both the client and server side, however, requires separate **SLAs** on both system sides. For simplicity of discussion, consideration will only be given to server-side evaluation hereinafter. Therefore, a test program or simulation is performed on the server side that simulates a query on database 105. In doing so, database 115 must first accept a login. This login is included in the simulation. The test program executes the login and database query, recording the startup time of database 115 and database 105. These startup times recorded from the test program are generally numerical in nature, and are subsequently converted to Boolean values through the aforedescribed comparisons to the associated **SLAs**. For this example, assume that on execution of the test program for a query to database 105, startup time for database 115 was recorded to be 1.25 seconds while subsequent startup time of database 105 was recorded to be 1.5 seconds. The access times, AT, of both would be recorded similar to that given below:

Detailed Description Text - DETX (18):

The associated **SLAs**, previously defined, are again given below:

Detailed Description Text - DETX (19):

Failure to meet the requirements of an **SLA** may be assigned a Boolean low, i.e., False or logical zero, and performances meeting the pertinent **SLA** being assigned a Boolean high, i.e., True or logical one. The numerical results of the test program may then be converted to Boolean attributes by comparisons to their respective associated **SLA** thresholds. In doing so, the test program results of the current example would respectively be assigned Boolean values as follows:

Detailed Description Text - DETX (22):

A final input is an original system model, upon which the system and method of the present invention builds, improving the accuracy and performance of the underlying system 100, illustrated in FIG. 1. It should be understood that the model of the IT system 100 is preferably developed such that it supports the

functions for which **SLAs** are defined. It should also be understood, however, that the model may be defined at any level and it is not necessary that the model be complete or consistent as is the case for expert systems. This is true due to the iterative adaptiveness of the overall system and method of the present invention in that over time the model automatically refines and corrects itself.

Detailed Description Text - DETX (23):

With the discussed inputs considered, the output of the system and method of the present invention may now be considered. Once sufficient historical data has been collected and stored in the database 315, data mining techniques familiar to those skilled interrogatory in the art may be applied to this collection of monitored data and its associated test data. Data mining techniques are then applied to these data and the various relations between the monitored system state data and the data on test performance success or failure are uncovered. These newly discovered relations are then used to update the existing IT model, thereby rendering the model adaptive. This unique feature of the present invention, i.e., its ability to adapt itself to the system it is used to **monitor** and model, enables the original model to be incomplete or inconsistent.

Detailed Description Text - DETX (24):

A decision tree algorithm is preferably utilized in the output where the Boolean value evaluated from the test program data and the corresponding **SLA** is used as the target attribute of the decision tree. Although decision tree induction methods are well known to those skilled in the art, FIG. 4 is provided herein to illustrate its usage. In operation, a targeted system component is selected, either by an administrator or autonomously, for analysis, and a decision tree 400 generated. This target component forms a root node 405 of decision tree 400.

Detailed Description Text - DETX (25):

The specific example illustrated in FIG. 4 shows a decision tree 400 for a query to the aforementioned database 115(B) of FIG. 1, where the performance of the query (QUERY_B) through the system 100 is targeted for analysis. The 50% noted at the target element 405 indicates that this target has been determined to be satisfied in 50% of the instances, i.e., the target **SLA** (access time less than or equal to one second) was satisfied half the time. The numerical value following the success percentage, i.e., 800, is simply an indication of the number of instances at which state data was recorded over a given time period. In other words, at this root level of analysis, in 800 queries of database 115, the aforedescribed target **SLA** of 1 second was met only half the time.

Detailed Description Text - DETX (26):

The branches of the decision tree 400 from the root node 405, i.e., an upper 410 and a lower 415 branch or element, also include monitored values and their determined relation to the performance success or failure of the target element 405. The upper element 410 of the first branch, for instance, indicates the effect of the number of network file server (NFS) daemons on the success or

failure of the target element 405. Branch 410 indicates that when the number of NFS daemons is greater than ten, the target element 405 (over a sample size of 350) was found to have acceptable performance 90% of the time. The evaluation of whether the target element 405 performance is acceptable is determined according to methods earlier discussed, specifically the methods of definition and evaluation of the performance thresholds or **SLAs**.

Detailed Description Text - DETX (27):

The lower branch 415 from the root node 405 indicates that when the number of NFS daemons is ten or less, the target element 405 (over a sample size of 450) has acceptable performance only 20% of the time. As shown in FIG. 4, the lower branch 415 is further split into sub-branches 420 and 425, denoting additional system attributes concerning the target element 405. Sub-branch 420 indicates that when the number of NFS daemons is less than or equal to ten and the number of logons to database 115 is greater than four, the performance of the target element 405 /database 115 (over a sample size of 20) is acceptable only 1% of the time, clearly demonstrating a system **resource** problem. The other sub-branch 425 indicates that when the number of NFS daemons is ten or less and the number of database logins is four or less, the target element 405 (over a sample size of 430) has acceptable performance 40% of the time.

Detailed Description Text - DETX (28):

Since the Boolean evaluation of the test programs are recorded in the historical database 315 shown in FIG. 3 with associated monitored system state data, and due to the Boolean values of the **SLA** parameters being used as target attributes in the decision tree, the decision tree 400 describes the influence of the **monitor** values, and thus system component states, on the target attributes. Factors on system component states that affect system performance the most appear close to the root node 405 of the tree 400. This can be seen in the example depicted in FIG. 4 where the first branch gives obvious indication of the most causal relations effecting performance of the target element 405.

Detailed Description Text - DETX (30):

As another example of the use of the aforedescribed decision trees, shown in FIG. 5 is a scatter diagram illustration of monitored values within the IT system 100 over time, particularly, the system access times to database 115. As is apparent from the diagram, although performance was good initially (most values at one second), over several weeks performance slowly decreased with most access times increasing to two, three and even four. Thus, the associated **SLA** for accessing database 115 is increasingly not met and an analysis of system performance is necessary to ascertain the source(s) of the problem.

Detailed Description Text - DETX (37):

As discussed, a number of benefits can be realized with the generation of the aforementioned decision trees, e.g., trend analysis for predicting future system failures and performing preventive maintenance. Performance optimization is readily apparent in reviewing the output of the decision trees, e.g., the increase in memory and daemon **resources**. It should be understood

that since parameters close to the root of the decision tree generally have the greatest influence on performance, different actions might be suggested to optimally influence those parameters. **Monitor** 310 optimization is another benefit that may be realized from the implementation of the principles of the present invention. Based on an analysis of the tree decisions, certain monitors 310 may be more or less relevant than other monitors with respect to a particular **SLA**. The positions of these monitors 310, or the **monitor's** frequency of data capture, could then be adjusted accordingly to facilitate a better analysis of the system 100.

Detailed Description Text - DETX (39):

In devising a proper monitoring scheme for querying database 105 or 115, it is apparent that monitors 310 taking system state information would be desired at least at user workstations 140 and 145, at which the queries may be made, a network hub 135, and the aforedescribed servers 110 and 120. State information would be desired at a minimum of these locations since all are directly involved in the path of required communication. With monitors placed at the aforementioned locations, it would be possible to define **SLAs** for both client- and server-side performance.

Detailed Description Text - DETX (40):

Furthermore, since one of the objects of the present invention is to uncover hidden or unexpected relations, a **monitor** 310 may also be placed at a printer 155, servicing the workstations 140 and 145, and synchronized with the test program of the **SLA** for querying database 115 or 105. Although it would not typically be expected for printer 155 to have any relation with the performance of workstation 140 or 145 users querying databases 115 or 105, the printer 155 is physically coupled to workstations 140 and 145, which themselves are coupled through the network 100 to the servers 110 and 120, as well as another server 160 and potentially many more components via the network hub 135. Such coupling can be seen to be a minimum requirement for functional interaction between various network 100 elements. Additionally, assume a network printer 165, servicing the network 100, is only online during certain hours of the day. During the hours in which the network printer 165 is online, it would be desirable to **monitor** state information of this printer to evaluate of the **SLA** related to querying databases 105 or 115.

Detailed Description Text - DETX (41):

Although the aforedescribed **SLAs** were defined with respect to the server side, inspection of FIG. 1 indicates why it is desirable that separate **SLAs** and corresponding test programs be defined additionally on the client side. For a client-side **SLA**, for instance an **SLA** for querying database 105 with the performance threshold defined as that time **measured** for startup of database 105 from initial user query, it is seen that these **SLAs** would not be identical. For this client-side **SLA**, it would be necessary to account for the delay encountered from the client-side workstation, either 140 or 145, through the hub 135 to the server 110. Since this communication path is not traversed when **measuring** from the server-side, it is reasonable to expect that the threshold on the client-side for this case to be slightly larger than the server-side threshold.

Detailed Description Text - DETX (42):
Furthermore, by having a client-side **SLA** related to the same function as an **SLA** defined to evaluate a server-side function, additional information may be recovered. In this example, by taking monitoring data on client-side information and having separate **SLAs** and separate test programs defined on the client side, information would be recovered that could determine relationships between the specified function, the involved servers and workstations, and the network hub 135. By defining and operating the test function solely server side, the same relations may be found as long as monitoring activity included workstation and hub states, but such relations may be determined more quickly by including **SLAs** and associated test programs both server and client side.

Detailed Description Text - DETX (43):
Consistent with the ongoing discussion, when all network 100 elements are functioning and **monitored, there are SLAs** defined client side and server side for the example database queries within the architecture depicted in FIG. 1. There will, therefore, be test programs launched client side and server side that simulate these queries from their respective sides of the network 100. Furthermore, these test programs are preferably synchronized with the aforementioned monitoring activities at the above-specified locations, which all constitute network 100 elements illustrated in FIG. 1.

Detailed Description Text - DETX (44):
The above, however, is not intended to suggest that, at execution of each defined **SLA** test program, state monitoring is performed at every available **monitor** 310. For instance, when the network printer 165 is taken offline at controlled and specified intervals, it is not necessary to take state information on this element when any test programs are executed. Furthermore, there would likely be network elements that are identified as physically (or otherwise) decoupled from those elements involved in certain functions. If such decoupled elements are properly identified, monitoring activity on these elements would not be necessary in the test program execution.

Detailed Description Text - DETX (45):
Throughout the discussion of the present invention, consideration has been given to essentially two functions and the development of thresholds **(SLAs), monitoring** activity, and analysis of such data. It should be apparent, however, that the present invention may include even more of such functions, with associated test programs, thresholds, associated synchronized element state monitoring, and subsequent analysis and model modification, as is understood by one skilled in the art.

Claims Text - CLTX (1):
1. In an information technology system having a multiplicity of interconnected nodes, a method for **optimizing** performance monitoring of said system, said method comprising the steps of:

Claims Text - CLTX (9):
(h) automatically modifying said steps of continuously monitoring and periodically collecting said system performance data at a plurality of said nodes, whereby said autonomous modification iteratively **optimizes** said continuous performance monitoring of said system; and

Claims Text - CLTX (11):
2. The method according to claim 1, wherein said steps (a)-(i) are repeated a plurality of times, whereby said continuous performance monitoring of said system is further **optimized**.

Claims Text - CLTX (13):
generating a test program pursuant to at least one **service level agreement,** said plurality of nodes for continuous monitoring and periodic performance data collection being selected pursuant to said at least one **service level agreement**.

Claims Text - CLTX (14):
4. The method according to claim 3, wherein said test program targets a target component within said system, said target component being selected from the group consisting of a system hardware **resource** and a system software application.

Claims Text - CLTX (27):
16. The method according to claim 15, wherein said system information comprises a plurality of **service level agreements**.

Claims Text - CLTX (37):
26. The method according to claim 25, wherein said at least one Boolean value corresponds to at least one **service level agreement** within said system.

Claims Text - CLTX (42):
monitor means, for continuously monitoring, at a plurality of said nodes, the performance of said system at the respective nodes;

Claims Text - CLTX (47):
modification means for automatically modifying said **monitor** and collection means for the continuous monitoring and periodic collection, respectively, of said system performance data; and

Claims Text - CLTX (50):
test program generation means for generating a test program pursuant to at least one **service level agreement**.

Claims Text - CLTX (53):
32. The system according to claim 31, wherein said target component is selected from the group consisting of a system hardware **resource** and a system software application.

Claims Text - CLTX (55):
34. The method according to claim 33, wherein said target component targeted by said test program is an underperforming system component, whereby said modification means automatically modifies the continuous monitoring and periodic collecting of said performance data on said underperforming system component by said **monitor** and collection means, respectively.

Claims Text - CLTX (56):
35. The system according to claim 28, wherein said modification means automatically modifies the periodicity of said performance data continuous monitoring and periodic collection by said **monitor** and collection means, respectively.

Claims Text - CLTX (66):
44. The system according to claim 43, wherein said system information comprises a plurality of **service level agreements**.

Claims Text - CLTX (76):
54. The system according to claim 53, wherein said at least one Boolean value corresponds to at least one **service level agreement** within said system.

Claims Text - CLTX (78):
56. An article of manufacture comprising a computer usable medium having computer readable program code means embodied thereon for **optimizing** performance monitoring of at least one node in an information technology system, the computer readable program code means in said article of manufacture comprising:

Claims Text - CLTX (87):
(h) automatically modifying said steps of continuously monitoring and periodically collecting said system performance data at a plurality of said nodes, whereby said autonomous modification iteratively **optimizes** said continuous performance monitoring of said system; and